Influence of the tie-break rule on the end-vertex problem

Pierre Charbit, Michel Habib, Antoine Mamcarz

Discrete Mathematics and Theoretical Computer Science DMTCS vol. **16:2**, 2014, 57–72

Influence of the tie-break rule on the end-vertex problem

Pierre Charbit* Michel Habib Antoine Mamcarz

LIAFA-CNRS, Université Paris Diderot-Paris 7, France
INRIA project-team GANG, France.

received 1st Nov. 2013, revised 11th Apr. 2014, accepted 27th June 2014.

End-vertices of a given graph search may have some nice properties, as for example it is well known that the last vertex of Lexicographic Breadth First Search (LBFS) in a chordal graph is simplicial, see Rose, Tarjan and Lueker 1976. Therefore it is interesting to consider if these vertices can be recognized in polynomial time or not, as first studied in Corneil, Köhler and Lanlignel 2010. A graph search is a mechanism for systematically visiting the vertices of a graph. At each step of a graph search, the key point is the choice of the next vertex to be explored. Graph searches only differ by this selection mechanism during which a tie-break rule is used. In this paper we study how the choice of the tie-break in case of equality during the search, for a given graph search including the classic ones such as BFS and DFS, can determine the complexity of the end-vertex problem. In particular we prove a counterintuitive NP-completeness result for Breadth First Search solving a problem raised in Corneil, Köhler and Lanlignel 2010.

Keywords: Graph algorithms, Graphs searches, end-vertex problem, BFS, DFS, LBFS, LDFS

1 Introduction

A search is a mechanism for systematically visiting the vertices of a graph. For graph searches, we will use the terminology and notation defined in [7] which are now quite standard, and we will consider that every graph search S will output its visiting-ordering σ, that is to say the order in which the vertices of the graph are visited by the algorithm. We will call σ an S-ordering. Considering a search will therefore mean here considering the set of all possible orderings produced by this search.

At each step of a graph search, the key point is the choice of the next vertex to be explored. Graph searches only differ by this selection mechanism during which a tie-break rule is used. As an example, for a Generic Search any neighbour of the already visited vertices is eligible. Any other graph search we are interested in (BFS, DFS, LDFS,...) is a specialisation of the Generic Search, and therefore inherits its behaviour with respect to connectivity. Since any generic search traverses the connected components of a graph one by one in any order, if we want to know whether or not a vertex t can be the last vertex of some search S on some graph G, it is enough to consider the end-vertex problem on the graph induced by the

*Partially supported by the French Agence Nationale de la Recherche under reference ANR-10-JCJC-HEREDIA

connected component of G that contains t. Therefore, in the following, $G = (V, E)$ will always denote a connected graph with n vertices and m edges.

In this paper we focus on the vertices that appear as the last vertex of various searches. Such a vertex often has nice properties with many uses in graph algorithms. Let us give a few examples.

- If the graph is chordal, any last vertex of a LBFS is simplicial and this is used to recognize chordal graphs in [16, 18]. Similarly for LDFS, since it is a MNS (Maximal Neighbourhood Search) we know from [17] that it provides a simplicial elimination scheme for chordal graphs.
- If the graph is a cocomparability graph, any last vertex of a LBFS, can be used as a source (resp. sink) in some transitive orientation of its complement (which is a comparability graph by definition).This result was first proved in [14]. This is used in a transitive orientation algorithm for comparability graphs [12].
- For arbitrary graphs, any last vertex of a LBFS belongs to a moplex [3]. Furtehrmore for arbitrary graphs, using [19] the last vertex of a LDFS belongs to a moplex M (a subset which is both a clique and a module). Furthermore the vertices of M appear consecutively in the LDFS ordering.
- For BFS, the last vertex is also important as it can be used as a heuristic to design fast algorithms for diameter computations [8].
- For DFS, we have no example of application in which finding the last vertex of a DFS is involved. But the question were asked in [6], and for sake of completion we have studied its complexity.

It seems natural to ask if one can recognize or compute the end-vertices of these searches. This problem was first studied in [6]. Let us state it more formally.

Beginning-end-vertex problem for a search S:
Input: A graph $G = (V, E)$, and 2 vertices s and t.
Question: Is there σ an S-ordering of V such that $\sigma(1) = s$ and $\sigma(n) = t$?

End-vertex problem for a search S:
Input: A graph $G = (V, E)$, and a vertex t.
Question: Is there σ an S-ordering of V such that $\sigma(n) = t$?

A graph is a *bipartite graph* if it contains no odd cycle. It is *weakly chordal* if it contains no induced cycle of length greater than four, and *chordal* if it contains no induced cycle of length greater than three. Alternatively, a graph is chordal if it is the intersection graph of subtrees of a tree. A chord xy in an even cycle C is odd, when the distance in C between x and y is odd. A graph is *strongly chordal* if it is chordal and every cycle of even length at least 6 has an odd chord. A graph G is a *split graph* if both G and $\overline{G}$ are chordal. A graph is a *path graph* if it is the intersection graph of paths in a tree.

In [6], it was proven that it is NP-complete to determine if a given vertex could be last in a LBFS-ordering. For basic searches such as BFS and DFS the question was left open. The purpose of this paper is to discuss the complexity of the end-vertex problem for several searches, and for several classes of graphs. In particular we will prove NP-completeness of the end-vertex problem for both BFS and DFS. The results are summed up in the following table. Results in bold typeface are from the present paper.

The proof of many NP-completeness results follow from the NP-completeness result for a subfamily of graphs. For example for DFS, we prove that the problem is NP-complete for strongly chordal split graphs, and from that it follows the same for all superclasses, like, split, chordal, weakly chordal and the class of all graphs. For unknown cases, we write in italic and between brackets our conjectures if any.

End-vertex results	BFS	LBFS	DFS	LDFS	MNS
All Graphs	**NPC**	NPC	**NPC**	**NPC**	?(*P*)
- Bipartite	**NPC**	?(*NPC*)	?(*NPC*)	?(*NPC*)	?(*P*)
- Weakly Chordal	**NPC**	NPC [6]	**NPC**	**NPC**	?(*P*)
- - Chordal	?(*NPC*)	?(*NPC*)	**NPC**	?	P [2]
- - - Split	**P**	**P**	**NPC**	**P**	P
- - - - Str.Chordal Split	**P**	**P**	**NPC**	**P**	P
- - - Path Graphs	?	?(*P*)	**NPC**	?	P

Even if a LDFS order is a special type of DFS, complexity results cannot be transfered from one to the other. Consider for example the class of split graphs, where the DFS end-vertex problem is NP-complete whereas the LDFS version is in P. Having said that, we cannot show that such a connection does not exist between BFS and LBFS.

The paper is divided in the following way. In the next section, we discuss the problem for generic search and generic layered search. Sections 3 and 4 are devoted to BFS, LBFS, DFS, LDFS. The last section contains the conclusion and a discussion on open problems. We will not always define precisely each of these searches. The reason for that is that we will use characterizations of points conditions of the orderings produced by these searches, called 'four points condition'. The reference for these results is [7], let us recall it here.

Theorem 1.1 (Four points conditions) *Let $G = (V, E)$ be a graph. Let $<_\sigma$ be a total order on the vertices of G. Define a triple of vertices (a, b, c) to be a characteristic triple if $a <_\sigma b <_\sigma c$, $ab \notin E$ and $ac \in E$.*

- *$<_\sigma$ is a BFS-ordering if and only if for every characteristic triple (a, b, c), there exists d such that $d <_\sigma a$, $db \in E$.*
- *$<_\sigma$ is a LBFS-ordering if and only if for every characteristic triple (a, b, c), there exists d such that $d <_\sigma a$, $db \in E$ and $dc \notin E$.*
- *$<_\sigma$ is a DFS-ordering if and only if for every characteristic triple (a, b, c), there exists d such that $a <_\sigma d <_\sigma b$ and $db \in E$.*
- *$<_\sigma$ is a LDFS-ordering if and only if for every characteristic triple (a, b, c), there exists d such that $a <_\sigma d <_\sigma b$, $db \in E$ and $dc \notin E$.*

We will use this theorem in this paper very often, and we will refer to it as the 'four points condition'.

2 Generic Search

Using the classification of graph searches defined in [7], the most basic graph search is the Generic Search. It is the search that at each step visits any vertex that has a previously visited neighbour. (See algorithm 1.):

Algorithm 1: Generic search(G,s)

```
S = {s};
for i = 1 ... n do
    if S does not contain any unnumbered vertex then
        Pick any unnumbered vertex v of V;
    else
        Pick any unnumbered vertex v of S;
    σ(i) = v;
    S = S ∪ N(v);
return(σ);
```

In this section, we will prove that the end-vertex and the beginning-end-vertex problems for the generic search can be solved in linear time. To do so, we will first give a characterization of the end-vertices of generic search:

Theorem 2.1 *A vertex t is the end-vertex of some generic search of $G = (V, E)$ if and only if t is not an articulation point (1-cutset) of G.*

We will need the following proposition:

Proposition 2.2 *Let $G = (V, E)$ be a connected graph. σ is a generic-ordering of G if and only if every vertex $\sigma(i), i > 1$ has a neighbour in $\sigma(1 \dots i-1)$.*

Proof: Immediate from the definition. □

Proof of Theorem 2.1: By assumption, we can assume that G is connected. Assume, by contradiction that t is an articulation point of G, and that there exists σ a generic-ordering of G such that t is last in σ. Consider C_1 the connected component of $G[V \backslash t]$ that contains $\sigma(1)$. Consider C_2 any other connected component of $G[V \backslash t]$ (since t is an articulation point of G, such a connected component always exists). Since G is connected by assumption, every vertex $\sigma(i), i > 1$ has a neighbour in $\sigma(1 \dots i-1)$. Consider c_2, the first vertex of C_2 in σ. $N(c_2) \subseteq C_2 \cup \{t\}$, but no vertex of C_2 appears before c_2, thus $t <_\sigma c_2$, a contradiction.

Conversely, assume that t is not an articulation point of G. Take any generic-ordering σ_0 of $G[V \backslash t]$, and consider $\sigma = \sigma_0.t$. We claim that σ is a legitimate generic-ordering of G. Indeed, $G[V \backslash t]$ is connected, so, since G is connected, every vertex $\sigma_0(i), i > 1$ has a neighbour in $\sigma_0(1 \dots i-1)$. Since G is connected, t also has a neighbour in $\sigma(1 \dots n-1)$, thus σ is a legitimate generic-ordering of G. □

Since articulation vertices can be computed in linear time [13], we immediately have:

Corollary 2.3 *The end-vertex and the beginning-end-vertex problems for the generic search can be solved in linear time.*

3 Layered Searches

A layered search is a search that at each step, selects a vertex among the unselected vertices that are at minimal distance from the source vertex. Hence, the end-vertices of such searches are simply the vertices that are furthest from some other vertex. Since it is polynomial to compute all pairs of distances, the end-vertex problem is polynomial for the generic layered search. Several implementations of layered searches, such as BFS or LBFS, have been introduced using different tie-break rules to choose the next vertex to be visited. In the following, we will see how these rules influence the complexity of the end-vertex problem.

3.1 Breadth First Search BFS

Here, we will follow the definition of BFS given in [11], that is a graph search in which the vertices that are eligible are managed with a queue (note that it differs for example from the definition given in [5], where BFS stands for what we call here layered search). This is the most common implementation of a layered search. The main result of this section is the fact that even for the class of bipartite graphs, the end-vertex problem for BFS is NP-complete. After that we prove that the same holds for weakly chordal graphs, that is graphs that do not contain any hole of length at least 5 (recall that a *hole* is an induced cycle). The main open question is about chordal graphs, so we prove at the end of this section that the problem is polynomial for a subclass of chordal graphs, called split graphs.

3.1.1 Bipartite graphs : NP-complete

Theorem 3.1 *Given a bipartite graph G and a vertex v of G, it is NP-complete to decide if there exists an execution of BFS on G such that v is the end-vertex.*

To prove Theorem 3.1, we will first prove the following theorem:

Theorem 3.2 *The beginning-end-vertex problem for BFS is NP-complete on bipartite graphs.*

To do so, we use a reduction from 3-SAT. To this purpose, we use a family of auxiliary graphs.

Definition 3.3 *For every $n \in \mathbb{N}$ we define a graph G_n, which has one special vertex r_n called the root. It is constructed recursively as follows :*

- *G_0 is the graph with one vertex r_0.*
- *G_n is constructed from G_{n-1} by first adding three vertices : the new root r_n, and its two neighbours y_n and $\overline{y_n}$, that are also adjacent to r_{n-1}. Finally we attach a path of $2n-1$ new vertices to y_n (respectively $\overline{y_n}$) and label its end-vertex x_n (respectively $\overline{x_n}$).*

The following properties are straightforward from the definition.

Proposition 3.4 *G_n is a bipartite graph that has $(2n+1)(n+1)$ vertices that all are at distance at most $2n$ from r_n. There are $2n+1$ vertices at distance exactly $2n$ from r_n, and these are $x_1, \overline{x_1}, x_2, \overline{x_2}, \ldots, x_n, \overline{x_n}$ and r_0.*

The following proposition is central to the reduction and concerns the order that we obtain on those $2n+1$ vertices when we do a BFS starting at the root.

Proposition 3.5 *Consider an order on the vertices of G_n given by an execution of a BFS starting at r_n. For each $1 \leq i \leq n$ at most one of x_i and $\overline{x_i}$ is before r_0. Moreover each of the 2^n choices of one among x_i and $\overline{x_i}$ for each i, can be obtained as the set of vertices that appear before r_0 for some BFS order of G_n.*

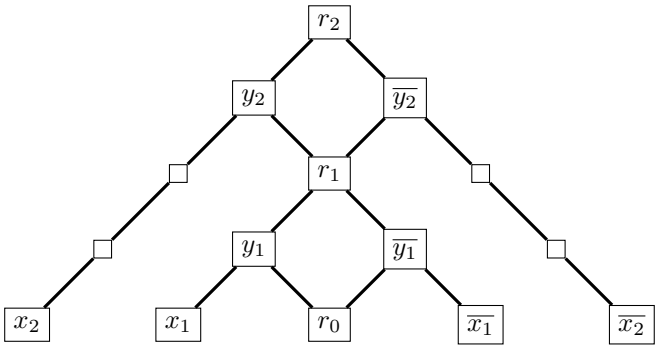

Fig. 1: The graph G_2.

Proof: We prove this by induction on n. For $n = 0$ there is nothing to prove. Assume the result is true for G_{n-1}. Starting at r_n, the BFS either continues with y_n then $\overline{y_n}$, or the converse. On the next layer, the vertices at distance two from r_n, there are three vertices : r_{n-1} and the two neighbours of y_n and $\overline{y_n}$ on their path of size $2n-1$ (call these y'_n and $\overline{y'_n}$). On this layer, it is easy to see that r_{n-1} can never be last. Moreover y'_n (or similarly $\overline{y'_n}$) is chosen after r_{n-1}, if and only if on the last layer of the graph, x_n is after any of the descendants of r_{n-1} (and in particular r_0). So the property we are trying to prove is true for $i = n$ and by choosing appropriately the order on these three elements of the second layer, we can decide to have either x_n or $\overline{x_x}$ before r_0 in the order. Note also that the order on the $2n-1$ descendants of r_{n-1} on the last layer, is exactly given by a BFS order on G_{n-1} so the induction hypothesis permits us to conclude the proof. □

We are now ready to prove Theorem 3.2.

Proof of Theorem 3.2: The proof uses a reduction from 3-SAT.
Given $F = (x_1 \dots x_n, c_1 \dots c_m)$ an instance of 3-SAT in which $x_i, 1 \leq i \leq n$ are boolean variables and $c_j, 1 \leq j \leq m$ are clauses. We construct a graph G_F by taking the graph G_n along with $m+1$ new vertices $c_1, \dots, c_m$ and t. Each vertex c_i corresponds to the clause of the same name in F and has neighbours in G_n exactly the vertices x_i or $\overline{x_i}$ that appear in it. Finally the vertex t has only one neighbour which is r_0. Note that these $m+1$ vertices are the only vertices at maximum distance $2n+1$ from the root r_n. We now claim that there exists a BFS execution on G_F starting at r_n and ending at t if and only if F is satisfiable. For any execution of BFS, we associate a truth assignment of the variables in F thanks to Proposition 3.5 as follows: we know that at most one of x_i and $\overline{x_i}$ is before r_0 in this order, and we choose this one to be the true literal. If both appear after r_0, then we choose arbitrarily one of the two. Note now that since t has only r_0 as a neighbour, a vertex c_i in the last layer will be before t in the order if and only if one of its neighbours (i.e. one of the literals appearing in it) was chosen before r_0 on the previous layer. In other words, c_i is before t if and only if the associated clause is true with the assignment described above, which proves the only if part of our claim. Conversely if F is satisfiable, then we use the second part of Proposition 3.5 to get a BFS that sorts all the true literals before r_0 among the vertices at distance $2n$. Then the previous observation on the c_i implies that t must be the last vertex, which concludes this proof. □

We are now ready to prove our main result.

Proof of Theorem 3.1: This proof uses a reduction from the beginning-end-vertex for BFS. Given $G = (V, E), s \in V, t \in V$, an instance of the beginning-end-vertex for BFS, we build G', an instance of the end-vertex problem by adding a pendant path p of size $d(G) + 1$ (where $d(G)$ is the diameter of G) to s. Now, every BFS that ends in t will have to begin at some vertex of p, and will visit s as the first vertex of G. □

In fact using a slightly more complicated construction, it is even possible to prove the following stronger result.

Theorem 3.6 *The end-vertex problem for BFS is NP-complete on bipartite graphs of maximum degree 3.*

As it is very similar to the proof of Theorem 3.1, we will not write the proof of this result, we just give the construction of the subcubic bipartite graph that gives the reduction to 3-SAT. For every $n \in \mathbb{N}$ we define a graph G'_n, which has one special vertex r_n called the root. It is constructed recursively as follows :

- G'_0 is the graph with one vertex r_0.
- G'_n is constructed from G'_{n-1} by first adding four vertices : the new root r_n, and its two neighbours y_n and $\overline{y_n}$. Both are also adjacent to the fourth vertex, r'_{n-1}. We add an edge between r_{n-1} and r'_{n-1}, and finally, we attach a path of $3n - 1$ new vertices to y_n (respectively $\overline{y_n}$) and label its end-vertex x_n (respectively $\overline{x_n}$).

Now let $F = (x_1 \dots x_n, c_1 \dots c_m)$ be an instance of 3-SAT in which $x_i, 1 \leq i \leq n$ are boolean variables and $c_j, 1 \leq j \leq m$ are clauses. For each literal x_i, let $c(x_i)$ be the number of clauses in which x_i appears.

Construct a graph G'_F the following way : start with graph G'_n defined above and to each vertex x_i of the construction, we attach a tree $T(x_i)$ such that: x_i is the root of the tree, $T(x_i)$ contains exactly $c(x_i)$ leaves that are all at distance $\lceil \log(m) \rceil$ from x_i, x_i has at most 2 children, and no vertex of $T(x_i)$ has a degree bigger than 3.

Finally, for every clause vertex c_j such that c_j contains the literal x, x', x'', we make the vertex that corresponds to c_j adjacent to one leaf of $T(x)$, one of $T(x')$ and one of $T(x'')$, in such a way that all the leaves of the $T(x_i)$ have degree 2, and we attach a path of length $log(c) + 1$ to r_0, and we call the other end of this path t.

Proof of Theorem 3.6: This new construction behaves similarly as the previous one and the proof directly follows. □

3.1.2 Weakly chordal graphs : NP-complete

Definition 3.7 *Let $G = (V, E)$ be a graph. The distance layers of a BFS of G that starts at s are the sets $L^s_1 \dots L^s_k$ such that $L^s_i = \{x | d_G(s, x) = i\}$, where $d_G(s, x)$ is the distance between s and x in G.*

Proposition 3.8 *A graph $G = (V, E)$ is weakly chordal if there exists $s \in V$ such that every distance layer of a BFS starting at s induces a clique of G.*

Proof: Since every distance layer of a BFS that starts at s induces a clique, no induced hole can contain more than two vertices of the same layer.

Moreover, no induced hole H of G can span over more than two distance layers of some BFS of G that starts at s. Assume by contradiction that H spans over $L^s_{i-1}, L^s_i, L^s_{i+1}$. H must contain 2 vertices of L_i, or else $H \cap L^s_i$ would be a 1-cutset of H (by definition there are no edges between any vertex of L^s_{i-c} and any vertex of $L^s_{i+c'}$, for any c and c' positive integers), but then H would contain a chord.

In this case, no induced hole of G can contain more than 4 vertices, and thus G is a weakly chordal graph. □

Definition 3.9 *Let $G = (V, E)$ be a graph, and let s be any vertex of G. The graph G^+_s is the graph G in which every distance layer of a BFS starting at s has been turned into a clique. More formally, $G^+_s = (V, E \cup E^+)$ with $E^+ = \{xy | xy \notin E \wedge d_G(s, x) = d_G(s, y)\}$.*

Lemma 3.10 *Let $G = (V, E)$ be a graph, and let σ be any permutation of V such that $\sigma(1) = s$. σ is a BFS-ordering of G if and only if σ is a BFS-ordering of G^+_s.*

Proof: First, assume that σ is a legitimate BFS-ordering of G. We will show that σ respects the four points condition for BFS on G^+_s. Consider a triple a, b, c such that $a <_\sigma b <_\sigma c$, $ac \in E \cup E^+$, and $ab \notin E \cup E^+$. $ab \notin E \cup E^+$ implies $ac \in E$. Indeed, the distance layers of any BFS appear consecutively in the BFS-order, so if ac was an edge of E^+, a, b, and c would belong to the same distance layer, which by definition, would imply that $ab \in E \cup E^+$.

In this case, the four points condition applied in G to the triple a, b, c implies the existence of $d <_\sigma a$ such that $bd \in E$, and so $bd \in E \cup E^+$.

Conversely, assume that σ is a legitimate BFS-ordering of G^+_s. We will show that σ respects the four points condition for BFS on G. Consider a triple a, b, c such that $a <_\sigma b <_\sigma c$, $ac \in E$, and $ab \notin E$.

Since σ respects the four points condition for BFS in G^+_s, we have to distinguish two cases.

1. $ab \in E^+$. In this case, by definition, a and b belong to the same distance layer. Since b is not the first vertex of σ, b must have a neighbour in G that belongs to the previous distance layer. In other words, there exists $d <_\sigma a$ such that $bd \in E$.

2. $ab \notin E^+$. In this case, since σ is a legitimate BFS-ordering of G^+_s, there exists $d <_\sigma a$ such that $bd \in E \cup E^+$. If $bd \in E$, we are done, so assume $bd \in E^+$. By definition, d, a, and b belong to the same distance layer, but then $ab \in E^+$, a contradiction.

□

Theorem 3.11 *The beginning-end-vertex problem for BFS on weakly chordal graphs is NP-complete.*

Proof: The proof uses a reduction from 3-SAT. Consider any instance F of 3-SAT, and let G_F be the graph defined in the proof of Theorem 3.2. Consider $G^+_{F_{r_n}}$. By Theorem 3.2, F is satisfiable if and only if there exists σ a BFS-ordering of G_F such that $\sigma(1) = r_n$ and $\sigma(n) = t$. By Lemma 3.10, the BFS-orderings of G_F that starts at r_n are the same as the BFS-orderings of $G^+_{F_{r_n}}$ that start at r_n. Since $G^+_{F_{r_n}}$ is weakly chordal by Proposition 3.8, we can conclude. □

Using the same construction as in Theorem 3.1 we have the following:

Corollary 3.12 *The end-vertex problem for BFS on weakly chordal graphs is NP-complete.*

Proof: Adding a pendant path to a vertex of $G^+_{F_{r_n}}$ cannot create any cycle. □

3.1.3 Split graphs : Polynomial

A graph $G = (V, E)$ is said to be split if and only if V can be partitioned into K, a clique, and S a stable set. We introduce first a definition and a notation.

Definition 3.13 *Let $G = (V, E)$ be a graph. We say that a vertex x dominates a vertex y if $N(y) \subsetneq N(x)$. For any $x \in V$ we denote by D_x the set of vertices dominated by x.*

Theorem 3.14 *Let $G = (V, E)$ be a graph, and let $t \in V$ be any vertex. A necessary condition for t to be the last vertex of some BFS-ordering σ of G is that there exists a neighbour x of t such that $D_t \subseteq N(x)$. If G is a split graph, then this condition is also sufficient.*

Proof:

First, assume that t is the last vertex of some BFS-ordering σ of G. Let x be the first neighbour of t to appear in σ. Let us prove that $D_t \subseteq N(x)$. Assume by contradiction that there exists $d \in D_t \cap \overline{N(x)}$. Since $d \in D_t$ it has no neighbour before x in σ. This implies clearly that d cannot be placed before x unless d is the first vertex of σ and x the second. But this is again not possible if d is not adjacent to x. So we can assume that $x <_\sigma d$, but then we can apply the four points condition to the triple x, d, t to find again a neighbour of d before x, giving again a contradiction since this vertex cannot be a neighbour of t.

Now, assume that G be a split graph and that there exists $x \in N(t)$ such that $D_t \subseteq N(x)$. Let (S, K) be split bipartition of $V(G)$.

First assume that there exists a vertex z in K that is neither t, nor any neighbour of t (in particular it implies that t is not in K and that its neighbourhood is included in K). Then one can start a valid LBFS by taking z, then all the other vertices of K that are not neighbours of t, then x, then all the remaining vertices of K (the other neighbours of t). Since we have just visited the vertices of a clique, this is a valid BFS start. After that the vertices that will be visited are necessarily vertices in S that have a neighbour which is not a neighbour of t. What remains is then all vertices of D_t plus t and possible twins of t. In any case these are all neighbours of x, which is their first neighbour in the order σ, which enables us to visit them in the order we like, and hence put t at the end.

If there exists no such z, then it means either t is already in K, or that it is adjacent to every vertex of K in which case we can add it to K. But now it means that every vertex in S is a neighour of x, so x is in fact universal. We can start a BFS in x and sort the other vertices in any order we like, in particular putting t at the end, giving us a valid BFS order. □

Corollary 3.15 *The end-vertex problem for BFS on split graphs can be solved in linear time.*

Proof: The condition of the previous theorem is clearly in linear time with usual graph algorithmic tricks. □

3.2 Lexicographic Breadth First Search LBFS

LBFS is a special kind of BFS introduced by Rose, Tarjan and Lueker [16]. It can be implemented in $O(n + m)$ time and has several very nice properties, especially with respect to its last vertex (see [1],[3],[12],[14],[18]).

In [6], it is proven that the end-vertex problem for LBFS is NP-complete for the class of all graphs, by proving it is NP-complete for the class of weakly chordal graphs. Again the frontier is for chordal graphs, and we were not able to prove a result for these graphs; as in the previous section, we prove that the problem is polynomial for split graphs.

3.2.1 Split graphs : polynomial

For LBFS, it was proven in [6] that the end-vertex problem is already NP-complete for weakly chordal graphs, and polynomial for interval graphs. A very interesting question is to find the border, and one important open problem is given by chordal graphs. As for BFS, we will prove that our problem is polynomial on the subclass of split graphs.

The following theorem of [2] gives a necessary and sufficient condition for a vertex to be last in a Maximal Neighbourhood Search (MNS) in a chordal graph.

Theorem 3.16 ([2]) *Let $G = (V, E)$ be a chordal graph. $t \in V$ is the last vertex of a MNS-ordering σ of G if and only if : t is simplicial, and the minimal separators of G included in $N(t)$ are totally ordered by inclusion.*

From this theorem, it is easy to derive an equivalent statement in the special case of split graphs.

Corollary 3.17 *Let $G = (V, E)$ be a split graph. $t \in V$ is the last vertex of some MNS-ordering σ of G if and only if*

1. *t is simplicial,*
2. *the neighbourhoods of the vertices of D_t are totally ordered by inclusion.*

In order to prove the polynomiality of our problem for LBFS we will in fact prove a very similar statement for LBFS.

Theorem 3.18 *Let $G = (V, E)$ be a split graph. $t \in V$ is the last vertex of some LBFS-ordering σ of G if and only if*

1. *t is simplicial,*
2. *the neighbourhoods of the vertices of D_t are totally ordered by inclusion,*
3. *the vertices of $V \setminus D_t$ dominate the vertices of D_t.*

With this theorem, it is easy to get the desired result.

Corollary 3.19 *The end-vertex problem for LBFS on split graphs is polynomially solvable.*

Proof: The simpliciality of t can be tested in linear time. Since the inclusion of 2 neighbourhood can be tested in $O(n)$, it is possible to check that the neighbourhood of some vertices are ordered by inclusion in $O(n^3)$. □

Proof of Theorem 3.18:

Note that since a split graph is chordal, the last vertex t of any LBFS order is simplicial. It is straightforward to check that in a split graph, this implies that there exists a bipartition (S, K), where $t \in S$. It implies in particular that $D_t \subset S$, and $N(t) \subset K$. This will always be the case in the rest of the proof.

Let us prove first that the 3 conditions imply that t is the last vertex of some LBFS-ordering of G. Assume $D_t = d_1 \dots d_k$, and assume $N(d_i) \subseteq N(d_{i+1})$ for all $i < k$. (By (2), such an order exists.) First, notice that $d_1, N(d_1), d_2, N(d_2) \backslash N(d_1), \dots d_k, N(d_k) \backslash N(d_{k-1})$ is a legitimate prefix of some LBFS-ordering of G. At this point, by condition (3), the labels of the vertices of $S \backslash D_t$ and the label of t will all be equal, and will contain the numbers of the vertices of $N(D_t)$. This will also be the case of the labels of the remaining vertices of K. Thus, it is possible to visit the vertices of $K \backslash N(t)$, then the vertices of $N(t) \backslash N(D_t)$. Since t does not dominate the vertices of $S \backslash D_t$, the vertices of $S \backslash D_t$ either have a private neighbour in K, or are twins of t. For the first case, the labels of those vertices will receive the number of their private neighbour in K before any other update of the label of t, i.e. the label of t will be smaller than the label of these vertices, and so t and its twins will be visited after them. It remains to visit the twins of t before t. Since $N(t)$ has already been visited, and since all these vertices belong to S, visiting a twin of t will not modify the label of any unvisited vertex, and since t and its twins have the same label at this point of the search, a LBFS can visit them in any order, including one that has t as end-vertex.

So now it remains to prove the converse direction, and thanks to Corollary 3.17 and the fact that every LBFS is a MNS, we only need to prove (3). Let us begin with two useful claims.

Claim 1 : there exists no triplet $a <_\sigma b <_\sigma c$ with $b \in S$, $c \in K$ such that $ab \notin E$ and $ac \in E$
Assume such a triplet exists. Thus, the four points condition implies the existence of $u <_\sigma a$ such that $ub \in E$ and $uc \notin E$. But since b is in S which is a stable set, it implies that u is in K which contradicts $uc \notin E$.

Claim 2 : there exists no triplet $a <_\sigma b <_\sigma c$ with $a \in S$, $b \in K$, and $c \in K$ such that $ab \notin E$ and $ac \in E$
Assume that such a triplet exists. Thus, the four points condition implies the existence of $u <_\sigma a$ such that $ub \in E$ and $uc \notin E$. Since $c \in K$, this implies that $u \in S$, which implies that $ua \notin E$. Now we get a contradiction by Claim 1 applied to (u, a, b).

Now we are able to finish the proof. Assume by contradiction that there exists $x \in S \backslash D_t$, $d \in D_t$, such that x does not dominate d. By definition of x, there exists $k \in K$ such that $k \in N(x) \setminus N(t)$. We choose k to be leftmost in σ with respect to this property. Define also $k' \in K$ such that $k' \in N(d) \setminus N(x)$. Finally let $k'' \in K$ be a vertex such that $k'' \in N(t) \setminus N(D_t)$, (recall that $N(d) \subsetneq N(t)$ by definition of D_t).

We prove a sequence or facts regarding the respective positions of these 6 vertices in the order σ.

- $d <_\sigma k''$: indeed if not, we can apply the four points condition on (k'', d, t) to get a neighbour of d, which is not a neighbour of t, contradiction.
- $d <_\sigma k$: previous fact plus Claim 1 applied to (k, d, k'')
- $k' <_\sigma k$: previous fact plus Claim 2 applied to (d, k, k')

- $k' <_\sigma x$: previous fact plus Claim 2 applied to (x, k', k)
- $k <_\sigma x$: previous fact plus Claim 1 applied to (k', x, k)

Eventually, we get that $k' <_\sigma k <_\sigma x$. But now we can apply again the four points conditions on (k', x, t) to get a vertex $k''' <_\sigma k'$ such that $k''' \in N(x) \setminus N(t)$, which contradicts the minimality of k. □

4 Depth First Search

By DFS we mean a graph search in which the eligible vertices are neighbors of the most recently visited vertex that still have unvisited neighbours. In other words, the eligible vertices are managed with a stack. We first study the standard DFS, here again we prove that our problem is NP-complete for various classes of graphs. Then in the second subsection we turn to LDFS, which was introduced by Corneil and Krueger in [7]. We prove that the problem for LDFS is NP-complete for weakly chordal graphs.

4.1 Depth First Search DFS

Theorem 4.1 *The end-vertex problem for DFS is NP-complete.*

Proof: The proof uses a reduction from hamiltonian path problem.

Given a graph $G = (V, E)$, we build G' an instance of the DFS end-vertex by adding a universal vertex u. We claim that G admits a hamiltonian path if and only if there exists a DFS of G' ending at u.

If G admits a hamiltonian path $p = v_1 \dots v_n$, then there exists T a DFS-tree of G that is a path. The path $v_1 \dots v_n, u$ is still a hamiltonian path of G', and is a DFS order of G' ending at u, which proves the if direction of our theorem.

Conversely, assume, by contradiction that G does not admit a hamiltonian path, but that u is still the end-point of some DFS-order σ of G', with DFS-tree T. Since G does not admit a hamiltonian path, every potential hamiltonian path of G' will contain u as an inner vertex, so T is not a path. Consider the leftmost leaf x of T. u comes after x, which means that u was not visited at the time x was. But in this case, u would have been added as a child of x, contradicting the fact that x was a leaf of T. □

Corollary 4.2 *The end-vertex problem for DFS is NP-complete on any graph class closed under universal vertex addition on which the hamiltonian path problem is NP-complete.*

Those classes include, among others the class of strongly chordal split graphs [15], and path graphs [4]. Moreover, since the hamiltonian path problem is NP-complete on planar graphs, and since adding one universal vertex to a planar graph turns it into an apex graph ([10]), the end-vertex problem for DFS is also NP-complete on apex graphs.

4.2 Lexicographic Depth First Search LDFS

Here we will prove that the end-vertex problem for LDFS is NP-complete even restricted to the class of weakly chordal graphs. As with BFS and LBFS, we leave the question for chordal graphs open, and as before, we prove that the problem is polynomially solvable on the subclass of split graphs.

4.2.1 Weakly chordal graphs : NP-complete

The proof will use a reduction from 3-SAT. Let I be an instance of 3-SAT. Assume, without loss of generality, that I contains at least 4 variables, and that no clauses contain a literal and its complement.

We build a graph G associated with I. The variables will be represented by M, the complement of a matching, with every non-edge standing for a literal and its complement. Add 2 non-adjacent vertices u and v, both universal to M. Let t be only adjacent to v. For each clause c_j, create a vertex adjacent to the literals c_j contains.

Proposition 4.3 *Any LDFS of G that starts at u must next visit a maximal (w.r.t. inclusion) clique K of M, such that $|K| = |M|/2$ and that K contains exactly one vertex from each literal and its complement, then v, then $M \backslash K$.*

Proof: Any LDFS of G that starts at u must next visit a maximal (w.r.t. inclusion) clique of $N(u)$. By construction, such a clique K is a subset of M. K cannot contain both a literal and its complement since their corresponding vertices are non-adjacent, and in order for K to be maximal, it must contain at least one vertex from each literal and its complement. Since $|K| \geq 4$, no clause vertex is universal to K, and the same holds for any remaining literal vertex by construction. Since v is universal to K, the label of v is larger than any other unvisited vertex of G, and so v must be visited at this point.

Since t is only adjacent to v, its label is smaller than the labels of the vertices of $M \backslash K$ (since they are all adjacent to v and u), so the next visited vertex (that must be a neighbour of v) is $x \in M \backslash K$. The remaining vertices of $M \backslash K$ will always have a larger label than the clause vertices, since they are all pairwise adjacent, and all adjacent to v, contrary to the clause vertices that may be adjacent to all the vertices of $M \backslash K$ that have been visited at some point, but that are non-adjacent to v by construction. □

Corollary 4.4 *For every truth assignment f, there exists σ a LDFS-ordering of V such that $\sigma(1) = u$ and that the vertices corresponding to the true literals of f appear after v in σ.*

Proof: Let F be the set of the vertices corresponding to the true literals of f, and $\overline{F}$ be the set of the vertices corresponding to the false literals of f. F and $\overline{F}$ both induce a clique of G, so there exists a LDFS of G such that $\overline{F} = K$. □

Proposition 4.5 *There exists a LDFS-ordering σ of V starting at u and ending in t if and only if f is satisfiable.*

Proof: Consider any LDFS-ordering σ of V with $\sigma(1) = u$ and $\sigma(n) = t$. We build a truth assignment I satisfying f by choosing for each variable x_i of f, the literals that appear after v to be the true literals. By Proposition 4.3, we know that for every literal l and its complement, at most one appears after v. If no literal corresponding to x_i appears after v, we assign any value to x_i. By Proposition 4.3, every clause vertex lies between v and t. In this case, by the four points condition, each triple v, c_i, t implies the existence of a neighbour of c_i between v and c_i, i.e. a true literal for the clause c_i.

Conversely, assume that there exists I a truth assignment that satisfies f. Let F be the set of the vertices corresponding to the true literals of I, and $\overline{F}$ be the set of the vertices corresponding to the false literals of I. By corollary 4.4 and proposition 4.3, there exists a LDFS of G that first visits $u, \overline{F}, v, F$. It only remains to prove that no c_i can appear after t in σ. Assume by contradiction that $c >_\sigma t$. Since I satisfies

f, c has a neighbour x between v and t. In this case, by the four points condition applied to the triple x, t, c, there exists a neighbour of t between t and x, a contradiction. □

Theorem 4.6 *The end-vertex problem for LDFS is NP-complete for weakly chordal graphs*

Proof: First, notice that in the graph constructed above for the reduction from 3-SAT, every clause vertex is simplicial (we can assume that the SAT instance does not contain a clause of the form $x_i \vee \overline{x_i}$ which is trivially satisfied). Notice that t is also simplicial since it is of degree 1. Since no simplicial vertex can be contained in an induced hole, the induced cycles of our construction are contained in the set of literal vertices, plus u and v, which induce the complement of a matching. Hence the only induced holes are of length at most 4, so the graph is weakly chordal. □

4.2.2 Split graphs : polynomial

In order to prove the desired complexity, we will in fact prove that the orders produced by LDFS on split graphs are the exact same as those produced by MNS searches (every LDFS is a MNS, but the converse is not always true). We will make use of Corollary 3.17 and we will prove the following theorem, which says that a vertex is the end-vertex of a LDFS if and only if it is the end-vertex of some MNS. Recall that $D_t = \{x \in V,\ N(x) \subsetneq N(t)\}$.

Theorem 4.7 *Let $G = (V, E)$ be a split graph. $t \in V$ is the last vertex of some LDFS-ordering σ of G if and only if*

1. *t is simplicial,*
2. *the neighbourhoods of the vertices of D_t are totally ordered by inclusion.*

Proof: Since every LDFS is a MNS, by Corollary 3.17, we only have to prove that these two conditions imply the fact that t is an end-vertex for some LDFS order. As in Section 3, note that t being simplicial, there exists a bipartition of $V(G)$ into a clique K and a stable set S such that $t \in S$, and for this bipartition, it is clear that every vertex of D_t (if any) is also in S. Denote by $(d_1, \ldots, d_k)$ the vertices of D_t, sorted by nondecreasing size of their neighbourhood. Let $D_1 = N(d_1)$, and $D_i = N(d_i) \setminus N(d_{i-1})$ for $i \geq 2$.

Consider the following order σ on the vertices of $K \cup D_t$ (as before when we write a set of vertices, like D_1, it means 'any order on the vertices of this set').

$$d_1\,,\, D_1\,,\, d_2\,,\, D_2\,,\, \ldots,\, d_k\,,\, D_k\,,\, N(t) \setminus N(D_t)\,,\, K \setminus N(t)$$

Let us check that this order is indeed a valid LDFS order for this subgraph, using the four points characterization. It is easy to check that there exists no triplet (a, b, c) of vertices such that $a <_\sigma b <_\sigma c$, $ac \in E$, and $ab \notin E$. Now at this point, the remaining vertices of G are the vertices of S that are not in D_t, that is t, its possible twins, and the vertices of S that have a neighbour in $K \setminus N(t)$. It is clear that the latter vertices have a larger label for LDFS than t or its twins, therefore we can put all of them first, then the twins of t, then finish with t to get a valid LDFS order σ. □

As before (see corollary 3.19), this clearly implies the desired polynomiality.

Theorem 4.8 *The end-vertex problem for LDFS is polynomial solvable for split graphs.*

5 Conclusion and Perspectives

We have proven a variety of results concerning the end-vertex problem for several graph searches, solving some problems raised in [6]. The main open question that remains is the complexity on chordal graphs of the end-vertex problem for BFS, LBFS, and LDFS.

Moreover, in [6], a very simple linear time algorithm for the end-vertex problem for LBFS on interval graphs is presented. We do not know if this extends to path graphs or cocomparability graphs (i.e. complements of comparability graphs). Since it is proven in [9] that for recognition LBFS behaves the same on interval graphs and comparability graphs, let us conjecture that the end-vertex problem for LBFS is polynomial on both classes.

It could also be interesting to find a class of graphs for which BFS and LBFS behave differently for the end-vertex problem.

Acknowledgements

The authors thank anonymous referrees for their careful readings and helpful remarks which help us to greatly improve the quality of this article.

References

[1] Pierre Aboulker, Pierre Charbit, Nicolas Trotignon, and Kristina Vuskovic. Vertex elimination orderings for hereditary graph classes. *Discrete math. to appear*, 2014.

[2] Anne Berry, Jean R. S. Blair, Jean Paul Bordat, and Geneviève Simonet. Graph extremities defined by search algorithms. *Algorithms*, 3(2), 2010.

[3] Anne Berry and Jean-Paul Bordat. Local lexbfs properties in an arbitrary graph. In *Proceedings of Journées Informatiques Messines*, 2000.

[4] Alan A. Bertossi and Maurizio A. Bonuccelli. Hamiltonian circuits in interval graph generalizations. *Inf. Process. Lett.*, 23(4):195–200, 1986.

[5] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms (3. ed.).* MIT Press, 2001.

[6] Derek G. Corneil, Ekkehard Köhler, and Jean-Marc Lanlignel. On end-vertices of lexicographic breadth first searches. *Discrete Applied Mathematics*, 158(5):434–443, 2010.

[7] Derek G. Corneil and Richard Krueger. A unified view of graph searching. *SIAM J. Discrete Math.*, 22(4):1259–1276, 2008.

[8] Pierluigi Crescenzi, Roberto Grossi, Michel Habib, Leonardo Lanzi, and Andrea Marino. On computing the diameter of real-world undirected graphs. *Theor. Comput. Sci.*, 514:84–95, 2013.

[9] Jérémie Dusart and Michel Habib. A new LBFS-based algorithm for cocomparability graph recognition. *Discrete Applied Math., submitted*, 2014.

[10] Michael R. Garey, David S. Johnson, and Larry J. Stockmeyer. Some simplified np-complete problems. In *STOC*, pages 47–63, 1974.

[11] Martin C. Golumbic. *Algorithmic Graph Theory and Perfect Graphs*, volume 57. Annals of Discrete Mathematics, 2004.

[12] Michel Habib, Ross M. McConnell, Chrsitophe Paul, and Laurent Viennot. LEXBFS and partition refinement, with applications to transitive orientation, interval graph recognition and consecutive ones testing. *Theor. Comput. Sci.*, 234(1-2):59–84, 2000.

[13] John E. Hopcroft and Robert E. Tarjan. Efficient algorithms for graph manipulation [h] (algorithm 447). *Commun. ACM*, 16(6):372–378, 1973.

[14] Norbert Korte and Rolf H. Möhring. An incremental linear-time algorithm for recognizing interval graphs. *SIAM J. Comput.*, 18(1):68–81, 1989.

[15] Haiko Müller. Hamiltonian circuits in chordal bipartite graphs. *Discrete Mathematics*, 156(1-3):291–298, 1996.

[16] Donald J. Rose, Robert E. Tarjan, and George S. Lueker. Algorithmic aspects of vertex elimination on graphs. *SIAM J. Comput.*, 5(2):266–283, 1976.

[17] Douglas R. Shier. Some aspects of perfect elimination orderings in chordal graphs. *Discrete Applied Mathematics*, 7:325–331, 1984.

[18] Robert E. Tarjan and Mihalis Yannakakis. Simple linear-time algorithms to test chordality of graphs, test acyclicity of hypergraphs, and selectively reduce acyclic hypergraphs. *SIAM J. Comput.*, 13(3):566–579, 1984.

[19] Shou-Jun Xu, Xianyue Li, and Ronghua Liang. Moplex orderings generated by the lexdfs algorithm. *Discrete Applied Mathematics*, 161(13-14):2189–2195, 2013.